Simulation of Automatic Canal Control Systems

A. J. Clemmens, M.ASCE[1]; E. Bautista, A.M.ASCE[2]; B. T. Wahlin, A.M.ASCE[3]; and R. J. Strand[4]

Abstract: Simulation models for unsteady open channel flows have been commercially available for more than 2 decades. Most of these models are now available for personal computers and can be used to study the control of irrigation canals. Studies on automatic control methods and algorithms have been performed on at least half a dozen of the available unsteady-flow simulation models. Although, many of these automation studies have been conducted by the institution that created the simulation model, these simulation models were not created with automatic gate control in mind, and thus one has to be intimately familiar with the source code in order to implement sophisticated control features. Three commercially available unsteady-flow simulation software packages that allow automatic control of gates based on algorithms written by users are: *CanalCAD* from the Univ. of Iowa, Hydraulics Lab; *Mike 11 version 3.2* from the Danish Hydraulic Institute; and Sobek from Delft Hydraulics. In this paper, we describe the various features of these unsteady-flow simulation packages and how they interface to control engineering software/code. There are a number of tradeoffs between simplicity and functionality. All these models present difficulties and have limitations. The hope is to provide guidance on the next generation of unsteady-flow canal simulation models so that control functions can be routinely applied.

DOI: 10.1061/(ASCE)0733-9437(2005)131:4(324)

CE Database subject headings: Unsteady flow; Open channels; Control systems; Simulation; Canals.

Introduction

Attempts to develop new canal control schemes during the 1980s were hampered by the difficulty in testing with the available unsteady-flow simulation models. For example, Burt (1983) and Zimbelman and Bedworth (1983) reported on the difficulties of using *USM* (Rogers and Merkley 1993), which at that time only ran on a mainframe computer. In the late 1980s, an ASCE task committee was formed to evaluate the various unsteady flow simulation models that were available for studying canal control methods. At that time, nearly all of these models were run on mainframe and minicomputers. By the time the task committee finished their work and published their results, many of these models were available for personal computers (Clemmens 1993). The committee was interested primarily in the models' ability to simulate water level and flow variations in canal systems with many gates and weirs. All of the available models adequately simulated water level response in these canals. A special issue of the *Journal of Irrigation and Drainage Engineering* (Clemmens 1993) presented the committee's findings, including descriptions of unsteady flow programs: *USM*, *CARIMA*, *CANAL*, *DUFLOW*, and *Modis*.

As a followup to the committee on unsteady-flow modeling, a new committee was formed to evaluate canal control algorithms. A wide variety of control algorithms had already been developed or were under development at the time (Malaterre et al. 1998). Typically, each algorithm was tested with a canal simulation model for which the investigator had access to the source code or to the source-code developers. The results of this task committee were presented in a special issue of the *Journal of Irrigation and Drainage Engineering* (Clemmens 1998). Examples of studies and the simulation models used to analyze control algorithms include: Malaterre (1998) who used *SIC*, Merkley and Walker (1991) who used *CANAL*, Liu et al. (1998) who used *CASIM*, Deltour and Sanfilippo (1998) who used *SIC*, Burt et al. (1998) who used *CanalCAD*, Lin and Manz (1992) who used *ICSS*, and Clemmens et al. (1997) who used *Mike 11*.

CanalCAD (Holly and Parrish 1992) was the first unsteady-flow simulation software that was developed primarily to test automatic canal-control algorithms. To date, only a few research groups have made use of this model (for example, Parrish and Khalsa 1997; Burt et al. 1998; and Wahlin and Clemmens 2002). *CanalCAD* is essentially a menu-driven front end to the unsteady simulation model *CARIMA* developed by *SOGREA* (see Holly and Parrish 1993). A *FORTRAN* subroutine "Custom" provides the interface through which the user can obtain water level information from *CanalCAD* simulation, execute control calculations, and pass gate position changes back to the simulator. The "Custom" subroutine can be modified and recompiled by the user, and thus, different control algorithms can be tested. Some simulation models, such as *DUFLOW* (Clemmens et al. 1993), do allow local control of individual gates, but do not allow a general control scheme to be implemented.

An interface between *MODIS* (Schuurmans 1993) and the *MATLAB* technical computing environment was developed to aid

[1]Director, U.S. Water Conservation Laboratory, USDA/ARS, 4331 E. Broadway Rd., Phoenix, AZ 85040. E-mail: bclemmens@uswcl.ars.ag.gov

[2]Research Hydraulic Engineer, U.S. Water Conservation Laboratory, USDA/ARS, Phoenix, AZ 85040. E-mail: ebautista@uswcl.ars.ag.gov

[3]Hydraulic Engineer, WEST Consultants, Tempe, AZ 85283. E-mail: bwahlin@westconsultants.com

[4]Electrical Engineer, U.S. Water Conservation Laboratory, USDA/ARS, Phoenix, AZ 85040. E-mail: bstrand@uswcl.ars.ag.gov

Note. Discussion open until January 1, 2006. Separate discussions must be submitted for individual papers. To extend the closing date by one month, a written request must be filed with the ASCE Managing Editor. The manuscript for this paper was submitted for review and possible publication on March 30, 2004; approved on August 19, 2004. This paper is part of the ***Journal of Irrigation and Drainage Engineering***, Vol. 131, No. 4, August 1, 2005. ©ASCE, ISSN 0733-9437/2005/4-324–335/$25.00.

in the implementation of control engineering features named *MODLAB* (Delft Univ. of Technology, unpublished). Unfortunately, *MODIS* was never developed into a commercial package and the effort was dropped.

In 1995, the Salt River Project (SRP) contracted with the United States Water Conservation Laboratory (USWCL) to study the potential for automating their canals. The SRP had previously contracted with the Danish Hydraulic Institute (DHI) to implement the *Mike 11* (DHI 1995) unsteady-flow simulation model on SRP's canal system. *Mike 11* did not have a built in user-written subroutine for canal operations. Instead, DHI provided the USWCL with information on how to access data on water levels and gate openings in memory. A user-written-code package was written in *Borland Pascal* to obtain data on water levels and gate positions, make control calculations, and modify the *Mike 11* solution coefficients by altering data in appropriate memory locations. This system was used to test canal automation on the upper Arizona Canal (Clemmens et al. 1997). The user-written code routines have since been changed to allow control of a user-defined canal network.

In 1993, Delft Hydraulics introduced a new unsteady-flow simulation software package, *Sobek*. It includes a link to *MATLAB* so that control decisions can be made within that framework. Water levels are passed to *MATLAB* and gate position changes are passed back to *Sobek*. The control routines are written as *MATLAB* "m" files. Recently, canal control studies have been conducted with the *Sobek–MATLAB* combination by Delft Hydraulics (van Overloop et al. 2003) and by the USWCL (Wahlin 2002).

Other unsteady-flow simulation packages are capable of implementing various canal control schemes. Our interest here is in examining the feature of those software packages that allow end users to write their own control routines without direct access to the source code. To our knowledge, this can only be done with the three software packages described above: *CanalCAD*, *Mike 11*, and *Sobek*. The purpose of this paper is to share our experiences with these unsteady-flow simulation software packages in testing canal control schemes, including useful features, current limitations, and future needs.

Canal Water Distribution Systems

The unsteady simulation engines discussed here were originally developed for river studies and later adapted to studies of man-made channels. Canal networks differ substantially from river networks in a number of important areas. First, canals branch out in the downstream direction while rivers converge. Second, unlike rivers, control structures are used in canals to control water levels and are extremely important model elements. Nearly all branch points are controlled by structures. Third, there is usually some level of control of inflow to the canal system, and to each branch, typically based on the cumulative water demand downstream. These demands play an extremely important role in determining how water is distributed and in evaluating the relative quality of that distribution.

The primary function of a control structure is to allow operators to divide the flow in the canal. Check structures are used for in-line regulation of water levels in a canal, while offtake structures control the flow of water to the head of offtake canals. Many canals have a check structure just downstream from each offtake. The idea is to adjust the check structure to control the water level upstream (head on the offtake gate) so that the discharge to the offtake is constant regardless of changes in flow in the continuing canal. Thus when a change in flow occurs in the continuing canal, the check structure is adjusted but the offtake structure is not. In practice, the offtake may be adjusted after the canal flow has stabilized. These operational practices often require some human judgment since canal check gate flow relationships are often not very accurate and the manual distribution of water is never perfect.

The normal practice for managing a given canal is to obtain a time series of water demands at various offtakes along the canal. This time series can be set by rigid policy or varied by daily farmer demands, depending on the amount of flexibility allowed. For most canals these demand changes are determined in advance or prescheduled. The job for canal operators is to route these flow changes through the canal to satisfy the demands at each offtake. For more flexible systems and in case of emergency, changes in offtake flow may occur without notice. Such unscheduled demand changes and errors in the setting of gates and flow rates require canal operators to periodically monitor the canal and to make gate changes to balance the inflow and outflow. Water levels upstream from check gates are common measures of their success in bringing the canal into balance. Most canal operators use water level targets at these locations, under the assumption that if the water level is correct then the flow through check and offtake gates will be correct. Such operation typically results in excesses or shortages at the tail end of the canal. The use of simulation models to improve canal control strategies and help solve this problem is the motivation behind studies on canal control.

Simulation of Unsteady Flow

In this section, we briefly discuss methods for modeling unsteady flow and provide details on how the three software packages (*CanalCAD*, *Mike 11*, and *Sobek*) solve the governing equations, boundary conditions, etc. Unsteady flow models are complex, even for well trained hydraulic engineers. An overview of these models is provided in ASCE (1993); their use is discussed in more detail by Burt and Gartrell (1993). Numerical methods for their solution are given in Strelkoff and Falvey (1993), while unique problems are discussed in Holly and Merkley (1993) and pitfalls are discussed by Contractor and Schuurmans (1993).

Unsteady flow in open channels is usually described by the one-dimensional Saint Venant equations, which are expressions for conservation of mass and momentum. These are hyperbolic nonlinear partial differential equations that cannot be solved analytically, except for a few simplified conditions. Thus all of the commercial unsteady simulation programs solve the governing equations numerically. Early models of unsteady canal flow used the method of characteristics. All of the models examined here use finite-difference methods, where the canal is broken into a series of cells, lengthwise. Nodes represent the boundaries between cells. More details on the various schemes discussed here can be found in Cunge et al. (1980) and Strelkoff and Falvey (1993).

CanalCAD uses the implicit Preissman scheme where all nodes are assigned values for both depth and discharge. The solution starts with known conditions at a given time and solves for the values at a future time. With N cells, there are $2N$ equations (continuity and momentum for each cell) and $2(N+1)$ unknowns (depth and discharge at each node). Two boundary conditions are required to solve the system of equations. Under subcritical flow, the boundary conditions are established at each

end of the canal using either one known parameter (i.e., a water level or a discharge) or a head–discharge relationship (i.e., one parameter defined as a function of the other). Typical upstream boundary conditions are a known inflow or a constant water level (reservoir), say upstream from a gate. In *CanalCAD*, the downstream boundary is represented by the condition chosen for the last reach of the canal (e.g., a gate or weir). The user can specify a water depth time series on the downstream side of this last structure. However, since this program was made for canal simulation, a normal-depth downstream boundary condition is not available. Efficient matrix solution schemes are available to solve these simultaneous equations. Interior check gate structures cause some difficulty for this solution scheme since there is a discontinuity in water depths. In *CARIMA*, extra nodes are required to separate the check structure calculations from the channel calculations. Also, multiple gates must be modeled as parallel branches in the canal that come back together. *CanalCAD* uses the computational engine of *CARIMA*, but it sets up the nodes for all gate structures so the user doesn't have to do it.

CanalCAD cannot handle branching canals, although the underlying *CARIMA* computational engine does. However, branching and looping through a reservoir is allowed. Neither *CanalCAD* nor *CARIMA* allow sections with supercritical flow. Such canal sections can be approximated by a straight drop followed by a canal section with a subcritical slope. Such an approach may not accurately predict transient behavior. In *CanalCAD*, the unsteady-flow computations must start from a steady-state initial condition. *CanalCAD* has separate steady-flow routines for determining these initial conditions.

Sobek uses the Delft Hydraulics Laboratory scheme to solve the Saint Venant equations, which is an implicit, staggered grid solution procedure (Stelling et al. 1998). The scheme is based on the concept of nodes where the water surface elevations are computed. These computational nodes are connected to adjacent nodes on the left and right through discharge equations. Thus, a computational grid is set up that consists of water surface elevations (h points) and flow rates (Q points). The h points are located at the computational nodes and the Q points are the connections between the nodes (Cunge et al. 1980). Like all implicit solutions, the Delft scheme advances from known conditions at one time to unknown conditions at a future time. *Mike 11* uses the implicit Abbott–Ionescu scheme (Abbott and Ionescu 1967) where every other node is a depth or a discharge (alternating). This differs from the Delft Hydraulics Laboratory scheme in that the Abbott–Ionescu scheme actually defines Q points as nodes, and so the system of equations is slightly different. As a result, the solution scheme is also different, even though they are similar in the way they have formulated the problem (i.e., both are fundamentally different from Preissman scheme). Both *Sobek* and *Mike 11* allow branching and looping. The branches emanate from h points, such that both branches have the same upstream water level.

The primary boundary conditions at both the upstream and downstream ends of the canal for both the Delft and Abbott–Ionescu schemes are known time series of water depth values. However, both *Sobek* and *Mike 11* allow the user to specify a discharge time series at the upstream and downstream ends rather than water levels. Any computational requirements for an upstream or downstream h node are handled internally by each program. Also, at the downstream end, a head–discharge relationship can be specified (e.g., based on normal depth). In our usual application of these two models, the downstream boundary condition is a fixed water level in a fictitious canal pool downstream from the formal end of the canal (e.g., on the downstream side of the last check gate). At check structures in *Mike 11*, the user can select the number and type of each structure. Any requirements for additional computational nodes or branches are handled internally by the software. In *Sobek*, the user must specify the structures as being on separate branches, diverging from a common upstream depth node and converging to a common downstream depth node. This adds complications in dealing with both input and output for these structures, but as discussed later, having a branch node downstream from structures can be useful.

As noted before, all three programs use what are considered implicit solution schemes. Such schemes simultaneously solve for depth and discharge at a future time based on known conditions at the current time. The size of the discrete time step (Δt) and distance step(s) (Δx) that are appropriate for these implicit schemes are determined by accuracy and numerical stability considerations (Strelkoff and Falvey 1993). High values of the Courant number usually lead to numerical instabilities, where the Courant number is $C_r=(|V|+c)\Delta t/\Delta x$, where V is the average velocity and c is wave celerity, $c=(gD)^{1/2}$, where g is acceleration of gravity and D is the hydraulic depth (area over top width). Numerical stability and accuracy are generally improved by using smaller time steps; however, at the cost of greater computation time and expense. *CanalCAD* and *Sobek* automatically reduce the time step when the Courant number gets too high. If the time step is temporarily reduced, it adjusts future time steps so that computations are made at times associated with the requested time step (e.g., for a 1 min time step, an extra computation at 7.5 min would create a computation at 8 min even if not required for stability). We did not find any evidence that *Mike 11* makes any automatic adjustments in the time step.

With implicit schemes, derivatives with respect to distance at the known and unknown time lines are weighted using a time weighting factor, whose value ranges between 0.5 and 1.0 (Cunge et al. 1980). This weighting factor, named θ, is applied to the future time, with $(1-\theta)$ applied to the current time. The value 0.5 for θ, a simple average of the spatial gradients, provides second order accuracy (i.e., gradients vary linearly for a second order system). For the Preissman scheme, setting the value for the time weighting factor equal to 0.5 makes the scheme nondissipative and can cause numerical instability and solution failure. Hence, a value of $\theta>0.5$ is commonly employed to dampen numerically induced oscillations. For rivers, a value of 0.55 is commonly used. *CanalCAD* uses a default value of $\theta=1.0$, which heavily weights the future value. Choosing a value of 1.0 helps assure that the numerical solution scheme will not fail, even in regions where the flow conditions are challenging for the solution scheme—but at the expense of not properly representing real conditions, i.e., real waves are also dampened. We typically use a value in the range of 0.6–0.67 as a compromise. *Sobek* also uses a default value of $\theta=1.0$, which leads to a stable, but damped solution.

The Abbott–Ionescu scheme in *Mike 11* uses several different time-weighting coefficients; a value of 0.5 (simple average) is used for most of the continuity and momentum terms, a coefficient δ is applied to the $\partial h/\partial x$ term in the momentum equation, and a parameter THETA is applied to the $\partial Q^2/\partial x$ in the momentum equation. The coefficient δ behaves similarly to the parameter θ, but applies only to one term. The parameter THETA provides time averaging for Q^2, but not in the same way as θ. The default value for δ is 0.5, a simple average. The default value for THETA is 1.0, which, as defined, gives a reasonable averaging over time (DHI 1995).

Cunge et al. (1980) show that for a Courant number of 1.0, the Preissman scheme with $\theta = 0.5$ accurately reproduces both the amplitude and phase of waves (pp. 86–89). For much higher Courant numbers, both the Preissman and Abbott–Ionescu schemes accurately predict the amplitude, but not the phase. However, the Abbott–Ionescu scheme even at a Courant number of 1.0 reproduces the proper amplitude, but not the phase. This results in a significant underprediction of celerity or wave speed. Our interpretation of their results suggest that the time step should be less than 20% of the time it takes for a wave to travel the length of a canal pool. We have experienced this problem in modeling the Arizona canal which has steep, short pools at the downstream end. This results in very short travel times for the waves in these pools. When the time step is chosen for conditions in average pools, these short, steep pools experience significant continuity errors and difficulty in arriving at a steady state. We have only been able to get reasonable results for this canal by significantly reducing the computational time step. We have not experienced this problem with *Sobek*, although a similar computational scheme is used.

Three aspects of *Sobek* and *Mike 11* make them particularly flexible, namely: (1) the initial conditions can be a dry canal; (2) canal beds can dry up; and (3) the flow can switch between subcritical and supercritical flow without causing any numerical problems (Stelling et al. 1998). The exact details of these schemes are proprietary; thus, they are not presented here. The usual approach in applying these models is to start with a steady-flow solution. *Mike 11* and *Sobek* can start from an established transient solution using output from a previous run. Establishing initial conditions is discussed in more detail below.

Check and Offtake Structures

Check structures are an extremely important aspect of the simulation of irrigation canals. Each of the programs discussed here uses different equations and procedures to compute weir and sluice-gate flows. Usually, the difference in equations between the programs is small and can be accounted for by small differences in gate openings for a given head and discharge. So while the actual gate openings might differ for different models, the general response of the canal to changes in flow and gate opening are not influenced significantly. An exception is gates that operate in the transition between free and submerged flow. Since the programs use different criteria for determining whether or not the gate is submerged, in this region there can be differences in response. Hopefully this is out of the normal range of operation and thus not a significant issue.

Since the flows at structures, and their derivatives, are part of the solution procedure, weir and gate equations are hard coded into these simulation packages. Because the Salt River Project wanted to model their radial gates, *Mike 11* made an exception for them and provided a means by which those gates could be modeled independently from the *Mike 11* source code. The hydraulics of these radial gates is now included as a selectable feature in *Mike 11* (*version 4*).

For modeling manual canal operations in any of these programs, the user must specify a time series of check gate positions. The check gate position for a particular desired steady-state flow rate has to be determined by inverting the gate equations. Such calculations have to be made outside the simulation software. In some cases, steady-state simulation can be used to obtain the needed gate openings as initial conditions for unsteady simulation. This will be discussed in the section on initial conditions. None of these models make this an easy task.

Gravity offtake structures are a key feature of irrigation canals. The three models discussed here allow a variety of ways to handle offtakes. All three allow a time series of outflow values, unaffected by canal water levels or flows. *CanalCAD* was developed for irrigation canal applications and is the only program that offers predefined options for simulating gravity offtakes. *CanalCAD* models offtakes as a simple submerged orifice. The user specifies the head-discharge exponent and a fixed downstream or tailwater level. The user also specifies a time series of offtake target discharges. Based on this schedule, the program internally computes the offtake gate opening for the specified upstream and downstream water level conditions. In general, the actual offtake discharge varies with the actual water level. *CanalCAD* offers three different zanjero turnouts, which primarily differ in how the discharge coefficient, which reflects the gate opening, is determined. In the first, the discharge coefficient is determined by the water level at the time of the flow change. Essentially, the assumption is that the zanjero (operator) sets the gate to the correct flow for the current water level. A significant limitation for this scenario is that the water level during these transient flow changes is frequently away from the desired or set point water level. By setting the gate to give the correct flow at this water level, the flow through the offtake will be in error if the canal water level is returned to its desired or setpoint value. If this happens, the zanjero will return to the site and reset the gate. To simulate the effects of the zanjero returning to reset the gate, the Irrigation Training and Research Center (San Luis Obispo, Calif.) zanjero turnout specifies a time at which the zanjero returns and resets the flow at the gate to match the desired flow for the water level at that time. In the third USWCL scenario, it is assumed that the offtake gate position is set so that the correct flow will result if the water level is at a user-specified target depth. This makes the offtake flow correct if the water level is returned to the setpoint. While it is unlikely that the zanjero would be able set the gate correctly in the first try based on future conditions, it is useful for simulating automatic controllers. In reality, offtake gate adjustments happen more often than suggested by any of these approaches.

Sobek and *Mike 11* are river models and do not provide preprogrammed options for modeling gravity offtakes. A time-series outlet flow can be specified in either model, but if one wants to model an outlet whose discharge varies with canal water level, the canal offtake must be modeled as a new canal branch. While this approach is ultimately more realistic than the simpler approach taken by *CanalCAD* (assuming such information is available), it adds a great deal of complexity to the canal topology. In *Sobek* and *Mike 11*, offtake gates are selected from the same options as for check gates.

All three programs allow some local automatic control features. At check structures, *CanalCAD* allows the selection of manual gates (time series of gate position), *FORTRAN* automatic gates (discussed below), Amil or Avis automatic gates, idealized gate, and *FORTRAN* idealized gates (discussed below). The Amil and Avis automatic gates are hydromechanical gates that approximately control upstream or downstream water levels (Rogers and Goussard 1998). *CanalCAD* attempts to model the response of these gates, including their decrement in control (i.e., small change in water level as a function of flow rate). For the idealized gates, the user can specify whether the gate maintains a constant flow rate or a constant upstream water level. *CanalCAD* computes the time series of gate positions to obtain the desired control.

However, there are cases when specifying a constant water level is physically impossible, and computational difficulties will result.

Sobek allows local automatic control of gates in four different modes: a time series (as with all models), a table of controls (e.g., gate positions or pump actions) as a function of water depth or discharge (called hydraulic control in *Sobek*), a so-called interval controller, and a proportional-integral derivative (PID) controller. The interval controller is similar to the little-man controller (Rogers et al. 1995). When the water level (or flow) is outside a given deadband, a fixed increment (interval) control action is taken. This is repeated at a given time interval. The PID controller is more sophisticated. It adjusts the position of a given gate based on errors in either a given water level or flow rate, also repeated at a constant time interval.

In *version 3.2*, *Mike 11* allows a table of gate openings to be entered by the user. This table can give gate opening as a function of time or as a function of depth or discharge at another location (similar to hydraulic control in *Sobek*). *Version 4* also allows several new features and logic-oriented selection of controls. New features include local PID control and a user-defined local control which allows iterative solution.

In parallel with *CanalCAD*'s *FORTRAN* automatic gates, *Sobek* and *Mike 11* allow user-written routines to control gate position based on water levels (discussed in the next section).

Initial Conditions

The development of initial steady-state conditions has proved one of the most troubling aspects of these unsteady-flow simulation models. All three of these models assume that they can approach steady flow conditions through unsteady-flow simulations where no changes in gate positions or boundary conditions occur. The initial conditions needed for the study of canal automation methods include initial water levels at the target values and initial gate positions for head gates, check gates, and offtake gates.

While backwater calculations with gradually varied, steady-flow equations could be used to determine initial steady conditions, none of these programs effectively use this approach. Use of unsteady-flow equations causes a number of complications. First, many canals have zero flow and a ponded water surface in the last canal pool. Second, for control purposes, we would typically like to set the water level on the upstream side of each gate to some desired value (setpoint). Establishing steady-state canal flows and the desired water levels upstream from gates through unsteady-flow simulation has proven to be difficult. Without additional constraints, roundoff error in canal inflows and outflow cause the canal to gain or lose water over time, making some desired steady conditions hard to approach. One pool will want to constantly drift away, typically the last one. We have used automatic upstream gates to develop a steady-state profile, where the gates move to keep the upstream water level constant. This provides a profile of water levels and the initial value for the check gates.

CanalCAD allows a steady-state condition to be determined prior to unsteady conditions. The unsteady flow equations are solved with no changes in boundary conditions until a steady condition results. However, any change in definitions nullifies the computed steady state. Thus one cannot choose one set of conditions to determine steady state (e.g., automatic upstream level control) and a different set of conditions to simulate control (e.g., *FORTRAN* automatic control). This is a significant limitation. While it ensures that the user does not have an in appropriate initial conditions, it does complicate the development of initial steady conditions. This limitation is imposed by *CanalCAD* only. The underlying *CARIMA* engine can use any initial condition, even unsteady flow. One trick we have used to establish the desired steady-state conditions has been to add a weir overfall in the last pool, with the weir height set at the desired water level. This is used to establish a steady state. Then at the start of the simulation, the weir crest is raised to the top of the canal. In this way, transients created during the unsteady approach to a steady condition can spill over the weir, while leaving the water level and volume essentially constant. Any minor error in inflow minus outflow due to roundoff errors can also spill over the weir.

In *Sobek* and *Mike 11*, we also determine steady-state conditions through unsteady simulation with no changes. However, in contrast to *CanalCAD*, there is no defined steady-state condition, so we have to force steady conditions by manually adjusting the boundary conditions so that there are no changes. *Mike 11* has a procedure for developing steady-state backwater curves, but we have not made this option work successfully for our application. *Sobek* and *Mike 11* allow the results of one simulation (in this case, a steady-state condition if we simulate long enough) to be stored and used as a starting condition file (hopefully a steady state). In *Mike 11*, these are known as hot-start files, in *Sobek* restart files. These files are the preferred approach to initiating an unsteady simulation test, and allow the same initial conditions for comparative testing. However in *Mike 11*, these files only include the water depths and flow rates at computational points. They do not include gate positions. This can be problematic since slight errors in these initial gate positions (typically determined from inverting the gate equations) can result in conditions that differ from the desired steady conditions (or at least conditions from the end of a prior simulation). With *Mike 11*, we have found it necessary to include a period of steady flow at the start of each simulation to reestablish a steady state. In *Sobek* this problem is avoided since the gate positions are included in the restart file, and these values can be read in *Matlab* prior to the first computations. This problem is also avoided with *CanalCAD*, but at the expense of making steady conditions more difficult to achieve and not allowing the same steady conditions for comparative testing. Even so, we recommend a short period of steady flow for each simulation prior to any testing.

We have found that it is sometimes difficult to have perfectly steady conditions at the start of a test, even though conditions were perfectly steady when the hot-start or restart files were established. Whether this is caused by slight differences in implementation of the gate hydraulic relationships or numerical issues is unclear. We often run an initial time to reestablish steady flows before initiating an unsteady test. Once one set of steady-flow conditions have been established for a canal, we have found it easiest to determine a new steady condition by simply making the change in inflows and outflows gradually and allowing the automatic controls to bring the system to a new steady condition over time.

In general when simulating irrigation canals, one cannot expect steady-state flow calculations to be trouble free with any of these programs. Some manipulation is nearly always needed to arrive at a stable steady condition. Unfortunately, it is difficult to predict when these programs will have difficulty and when they will not. A zero-flow condition at the end of a canal is particularly unforgiving. As discussed earlier, we have experienced computational problems with *Mike 11* at the end of long canal systems,

where water levels and volumes appear to drift, apparently in violation of continuity.

Operating Systems and Programming Languages

CanalCAD, the underlying computational engine *CARIMA*, and the subroutines for control are all written in *FORTRAN*. *CanalCAD* is a DOS application, but has a relatively friendly interface. It has been upgraded to operate under *Windows NT* (and all levels inbetween). We have had *CanalCad* working in *Windows 2000* and *XP*, but we have not been able to do that consistently and are unsure of the settings and versions of ancillary software required. The system is compiled with *Visual Fortran*. Object codes for all *CanalCAD* and *CARIMA* routines are included. The user written code routines for canal control can be edited, compiled, linked with the *CARIMA* object code, and run, all within *CanalCAD*. This works relatively seamlessly, although debugging the control routines is not convenient in this configuration.

Mike 11, *version 3.2*, was developed for DOS and Unix environments, and represents one in a series of software products stemming back to 1972 (DHI 1995). This version of *Mike 11* runs in operating systems up through *Windows 3.1*, *95*, *98*, and *NT*. *Mike 11* has a newer version 4 that operates in *Windows 2000* and higher versions. The ability to include user-written codes to automatically control canal flows is now supported in version 4. The DHI allowed us, through cooperation with the Salt River Project, to have access to the data structures of *Mike 11 version 3.2* through routines written in *Borland Pascal version 7* (for details see Clemmens et al. 1997). The user-written control subroutine is compiled and linked with *Mike 11* off line.

Sobek was developed for the *Windows* environment. It can operate under the *Windows 95/98*, *Windows NT 4.0*, *Windows 2000*, and *Windows XP* operating systems. User defined control routines are written either in *MATLAB* (*MATLAB users guide* 2000), a technical computing environment or with rule-based methods within the real-time control (RTC) module. We have not used these routines, and so here only discuss the use of *Matlab*. During simulation, *MATLAB* runs in parallel with *Sobek*. When *Sobek* runs a simulation, it starts *MATLAB* and causes *MATLAB* to run the control "m" file. At each control time step, *Sobek* passes control to *MATLAB*, which then executes the m file and passes control back to *Sobek*.

Debugging errors in the controller code can be a problem for these programs. In *CanalCAD*, a crash in the control code will cause the simulation to abort. Since *CanalCAD* runs in a DOS shell, brute-force debugging with write statements is the only useful option. The execution subwindow within *CanalCAD* does not have scrolling capabilities. With the *Sobek/MATLAB* combination, debugging the control code is also difficult. During *Sobek* simulation, a *MATLAB* window does not automatically show up. While standard *MATLAB* debugging can be used, if the control routine in *MATLAB* fails, *Sobek* continues without it. The program beeps to let you know there is a problem, which cannot be heard unless the speaker volume is sufficiently loud. The *MATLAB* code essentially has to be debugged separately by assigning "dummy" water levels rather than values supplied by *Sobek*. With all three programs, debugging was performed by writing output to a file for later interrogation. Standard debuggers are not very effective for programs with a dynamically allocated structure, such as the user-written code interface to *Mike 11*.

Data Input and Output

CanalCAD is set up to have data on profiles and cross sections entered from the keyboard. It is meant for modeling simple systems with a minimum level of complexity. As such, *CanalCAD* is very useful for theoretical studies and for teaching. One can enter data on a new simple canal quickly and easily in *CanalCAD* with a minimum of training. The canal starts with one pool and pools are added by the user. Each pool has a check gate at the downstream end. The pool starts with one reach with a defined length, slope, cross section, roughness, etc. Alternatively, the elevation at the downstream end of the reach can be given and the slope computed from the elevation of the last reach minus the drop in elevation between reaches. The cross sections are defined at the end of a reach, but apply to the entire reach. Flow resistance is computed with the *Manning* n, the same value for the entire reach. This is consistent with the way canals are designed and built. *CanalCAD* defines location with chainage or distance from the head, starting at 0+00. However, it also allows the user to define reach length. Changing the length of one reach correspondingly alters the chainage of all downstream reaches. Within each reach, the user can add turnouts, weirs, culverts, reservoir outlets, etc. These items are added within the reach dimensions. Canal definitions are saved in binary *CanalCAD* files.

CanalCAD provides graphical and tabular output at user-defined locations, on screen. Tabular output can be saved as a text file. The specific data elements to be output are specified by the user, and this selection remains until changed. This has proven to be very useful since the results of several runs can be quickly output in the same format without having to reselect data elements. Graphical output gives one variable per graph and is automatically scaled by *CanalCAD*, which we have found to make all graphs readable. Not all computational results are available to the user, only those at locations deemed useful for studying canal control behavior. This reduces the complexity of selecting output. For theoretical studies of open channel flow this may be a disadvantage, but for studies of canal control this has proven very useful. User names for structures are not allowed in *CanalCAD*, so the user has to keep track of pool/structure numbers.

Mike 11 was developed during the era of batch input data (text) files and has maintained that approach for storing data, even though data input can be provided through clumsy *Windows* interfaces. (The newer version, not discussed here, is much more friendly.) This is similar to the United States Army Corp of Engineers Hydraulic Engineering Center (HEC) formatting of data. The HEC format has become somewhat of a de facto standard for defining river and canal profiles and cross sections in the United States. If someone has modeled the system before, they likely have HEC files. In keeping with this approach, *Mike 11* has separate files for topology, boundary conditions, cross sections, time series, etc., all of which can be mixed and matched, although in our current version, special procedures were developed to import HEC data and put it into *Mike 11* format. We assume that this conversion is available in the current *Mike 11* version. In *Mike 11*, cross sections are defined at specific chainages—or distances from the head of the canal or river. This is a common river-modeling format. The topology essentially links defined cross sections. Slopes result from the elevation difference between cross sections. If cross-section information is needed at intermediate points, it is determined through interpolation. The user can choose from among several methods of computing flow resistance (e.g., *Chezy C*, *Manning* n, etc.). The standard method for entering these data is to assign a resistance

value for each cross section. Interpolation is used to determine the hydraulic conveyance for computational grid points and cells between defined cross sections. Structures are located by chainage and river segment. If the alignment of a reach changes such that it's length changes, then the chainages for all cross sections downstream need to be changed. For *Mike 11*, the cross sections can be exported to text files, the chainages edited, and then the files read back into *Mike 11*. This all provides the user with a great deal of flexibility. Users familiar with HEC programs will find them easier to use than the other programs discussed here. However, it should be noted that conversion from Hydraulic Engineering Center river analysis systems (HEC–RAS) files is not necessarily straightforward. Testing should be done to assure that "similar" hydraulic performance is achieved (e.g., under steady conditions).

Mike 11 provides graphical and tabular outputs. Flow level and discharge hydrographs are generated at locations that are manually selected by the user. The user can select any user- or program-defined calculation points (H or Q) to display these results. Selection is based on location (chainage) and not a particular labeling scheme. *Mike 11* allows the user to create and save lists of output items and, therefore, results can be easily presented from one simulation to the next for a given project, once the lists are created. Furthermore, like many of the *Mike 11* files, the output lists can be easily created and edited as text files outside the *Mike 11* environment. Graphical and text outputs can be displayed on screen, or can be saved to, respectively, graphical or text files. Graphical files are saved in the *Windows* Metafile format. Hydrographs are plotted individually, but multiple plots can be included in a page. Graph scaling parameters can be defined automatically or manually. The program also allows other graphical options to be modified. A limitation of the text output for *version 3.2* is that it is limited to five time series per screen, so additional time series are appended to the bottom of the file. Hence, some manipulation is needed to transfer this text output to a spreadsheet.

Sobek was designed to operate in a geographical information system (GIS) context. River or canal topology, cross sections, and structures can be exported from GIS and then imported into *Sobek* (see user's manual for specific formatting information). Nodes are defined at specific $x-y$ coordinates and connected in a graphical environment, after which cross sections are defined and structures are added. This can be done by essentially tracing over a map. It assumes a straight line between cross sections. Flow resistance, conveyance, and interpolation between cross sections are defined with the same approach used by *Mike 11*. However, if the alignment results in a greater length than a straight line, then new nodes need to be defined at intermediate points that more correctly define the alignment. This does not require new cross section definitions, only new node points. In *Sobek*, errors in coordinates are difficult to correct since data files cannot be exported. For theoretical studies, this has proven problematic since it is not easy to modify a reach once it has been defined (i.e., while nodes can be moved in "vector" mode; cross sections, structures, etc. do not move with them). Cross-section information can be exported for editing with *Microsoft Excel*. *Sobek* can also read HEC cross-section files as a basic data input. For new studies, the GIS format has some real advantages.

Output from *Sobek* on screen is primarily graphical. *Sobek* allows a wide variety of data to be graphed and many similar data elements can be shown on the same graph. Automatic scaling often makes these graphs hard to read, particularly when one data element is a straight line at the top or bottom of the graph, where it essentially disappears. Tabular data are not available on screen. The tabular data can be saved in a variety of formats (including comma-delimited *Excel* files). In the tabular data, the data elements to be saved must be selected each time, and they are selected by groupings. Further, one has to sift through the large number of data elements to select those that are of use in canal control studies. The case analysis tools can help to organize this data output for a given case, but we have not fully utilized this feature and so do not know its limitations. The output information is presented in terms of the identity (ID) numbers assigned by *Sobek*, unless overridden when created. Limited editing of these ID numbers is possible, but the options are not particularly useful. Thus, it is up to the user to associate an ID number with the appropriate gate position, water level, or flow rate. If the nodes are not given useful names by the user, the process is cumbersome. And even though *Sobek* assigns numbers to nodes, they are treated like characters, so that for example 49 and 499 precede 5. Nodes can be given names and the names edited, but these are not displayed in the output list, so the names cannot be used to select structures of interest.

Computational Grid Points

These programs differ in the way computational grid points are located and the options available for changing them. The computational grid points needed in the computational engine for the reach segment are determined by *CanalCAD* automatically. The user has no control over their spacing and location. Since the same cross section is used within each reach, no interpolation of cross section information is needed. *CanalCAD* provides grid points on both sides of structures, such that the hydraulics of the structures can be accurately modeled. These grid points on either side of the structure are actually branch points in the computational scheme, and multiple gates at a site are treated like separate branches. This allows the computational grid spacing for the canal reaches to be distinct and separate from the short distances between grid points around the structures. We have not experienced any computational difficulties with *CanalCAD*, so to date the automatic routines within *CanalCAD* appear to work well.

In *Mike 11*, the topology is defined first by creating a database with cross-sectional information for all river branches. A particular river branch is defined by the endpoints chosen from the cross-section database. Thus at least two cross sections are defined for each branch. The location of each cross section is defined by the chainage (distance along the river branch) for each associated river branch. All of these cross sections represent h-calculation points. *Mike 11* automatically inserts additional cross-section points as needed by linear interpolation based on the user-defined maximum distance between h points. The solution scheme requires alternating Q and h points, so *Mike 11* automatically adds Q points as needed. The user can add structures as well as point inflow or outflows. These constitute Q points. If the addition of these user-defined Q points requires addition of h points, *Mike 11* adds them automatically.

One problem we have experienced is the need for additional h points on the downstream side of structures. For long reaches, the first h point downstream from a structure can be hundreds of meters downstream. This may provide unrealistic estimates of backwater on the structure and lead to inaccurate prediction of gate discharge. (This is not an issue for free-flowing orifices or weirs). In some cases, we have added an additional computational

point close to the downstream side of the structure (e.g., 30 m). Unfortunately, the addition of this closely spaced node downstream from structures has caused problems with the computation scheme. In order to get stable results, we have been forced to reduce the computational time step.

A second problem we have experienced is associated with point inflows and outflows (e.g., an idealized offtake with no canal branch). If this point outflow is too close to a structure that we are controlling automatically, we have been forced to put additional Q and h nodes between them. In our scheme, we use the upstream discharge to compute the gate position for the given water levels. The value from the point outflow node is the value on the upstream side of the outflow, while our scheme requires the downstream value. This problem is avoided by either placing the point outflow further upstream or by using a real offtake, i.e., modeled as an actual canal branch. In general, anything that forces us to use closely spaced nodes causes us to use smaller time steps.

In *Sobek* the main nodes that define the boundaries between reaches are established first. Then these nodes are connected to form reaches, and finally cross sections, structures, and computational nodes are placed on the reaches. This is done reach by reach. The user can specify the spacing of computational nodes for each reach individually, or for the entire network. Computational nodes can be added and deleted at any location on a reach, as decided by the user. For canal control applications, it has proven convenient to place a main node downstream from check structures where the downstream level influences the gate discharge. This forces the calculation of water surface elevations there and separates the main canal reach from the reach that contains the structure. This is similar to the approach taken in *CanalCAD*, in which a branch point is normally defined just upstream from the check structure where the offtake (branch) is located. Then another branch point (connection node) is added just downstream, even though the canal does not branch there. Where this was not done, we used a relatively narrow computational grid spacing for reaches where gate submergence was an issue. To date, we have not had computational distance-step problems in *Sobek* with either of these approaches. (The addition of a downstream branch point may solve the problems experienced with *Mike 11*, however we did not do that in any of our studies).

Interface to Control Logic

Each program has a unique method for allowing the user to interface control calculations with the simulation of water flow. In general, these calculations require information from the simulation program on current water levels, gate positions, etc. The control program then determines needed changes in gate positions and passes this information back to the simulation program. To our knowledge the three programs presented here are the only ones that allow any user to have this functionality. These are discussed in the order in which these control features became available.

CanalCAD

CanalCAD was developed specifically to provide a software environment for testing canal control algorithms. At each time step, *CanalCAD* makes a call to *FORTRAN* subroutine "custom." This call is made once for each gate that is under automatic control. The user specifies the name of the file (e.g., auto.for) that contains this subroutine plus any subroutines called by custom. This file is selected by the user in the "Input" menu under "Control" from available "*.for" files on the *CanalCAD* directory. This file can be selected from a directory list and edited from this menu without leaving *CanalCAD*. Further, once this menu is exited, the user's control file is recompiled and linked to *CanalCAD*'s object code with *Digital Visual FORTRAN* (*version V6.0A*). Compiler errors will show up on a DOS screen within *CanalCAD*.

The *CanalCAD* shell essentially sets up input data for the *CARIMA* computational engine "C2." Once simulation starts, this old *FORTRAN* engine has control, communicating with the "custom" subroutine. At the completion of the simulation, CPU control is passed back to *CanalCAD*, along with the requested data describing the results of the simulation.

The "custom" subroutine has a series of data elements that are passed from *CanalCAD* to "custom." These data elements represent data for the specific check gate for which control is to be passed back to *CanalCAD*. Canal pools and gates are numbered consecutively starting at 1, with no user-defined names allowed. The original assumption behind *CanalCAD* was a series of local control devices—one controller for each gate based on local water levels (pools upstream and downstream). In 1995, a series of function calls were provided that allow access to data for the entire canal, including; water levels, gate positions, turnout flow schedules, pool volumes, etc. This allowed the application of centralized control logic. In our current applications, we compute all control actions (either gate-position or flow-rate change) the first time "custom" is called at a time step. These requested control actions are saved, and when "C2" calls "custom" for each gate, the necessary gate movement values are passed back to "C2." If the control action is a flow-rate change, we then compute the necessary gate movement by inverting the gate equation. Just prior to passing the value of gate position change back to "C2," we implement any gate-movement constraints. Since "C2" passes the time to "custom" it is relatively easy to determine the first time "custom" is called for a given time step.

After the last automatic control gate is called, the changes in gate position are implemented in the simulation routines within "C2" and the computations are started for the next time step. The simulation and control parts of *CanalCAD* operate as one executable program. Debugging and error checking are not particularly convenient in this environment because error messages can only be written to a DOS window within *CanalCAD* or to an external text file. The scrolling and overwriting features of *CanalCAD*'s DOS window make it very difficult to read and interpret the output, and thus we recommend writing debugging output to a text file.

In order to control flow rates at check gates, the gate equations used by "C2" must be programmed into "custom." The user is responsible for determining an appropriate gate opening, including deciding whether the gate is free flowing or submerged, in or out of the water, etc.

Overall, we have found this control interface to be relatively straightforward and effective. The user is entirely responsible for writing the control logic. However, we have found that simple control logic can be easily programmed in this environment. Debugging is not particularly convenient in the DOS environment imbedded in *CanalCAD*. Our impression is that it is very fast relative to *Mike 11* and *Sobek*, although controlled, timed tests have not been run.

Mike 11

Mike 11 allows a special interface to access information from its computational routines. This interface is known as the user-written code (*UWC*) (see DHI 1995 for details). Both *Mike 11* and the *UWC* are written in the *Borland Pascal v7.0* programming language. The main computational module in *Mike 11 version 3.2*, "MIMIC620," calls the *UWC*. These calls are made, first, to initialize the data associated with the *UWC* structures and, later, to carry out the hydraulic computations. *MIMIC620* passes three pointer arguments to the *UWC* that define where *Mike 11* data are stored in memory. These three pointers are associated with the physical control structures (*weirstruc*), the computational grid points (*gridstruc*), and other model variables (*mm_type*). Access to memory location is through linked lists.

In order to access the proper data, the *UWC* must map data elements associated with canal structure identically to *Mike 11*, byte by byte. Essentially the *UWC* has to repeat some organizational code that already exists within *Mike 11*. Early releases of version 4 did not support the *UWC*, however current releases fully support the *UWC*.

However, once the *UWC* mapping has been established, the code has complete access to all *Mike 11* data associated with structures and grid points. Unfortunately, information on time and iterations were not provided until *version 4*. *Mike 11* calls the *UWC* once for each *UWC* structure at each iteration in the computation, rather than at the end of the time step. This seems an unnecessary complication and makes the *UWC* overly complex. In addition, the sequence in which the *UWC* sites are processed is the same each time but does not appear to follow a logical pattern, and the sequence changes if the topology is modified. The *UWC* is also responsible for any naming of gates and water levels, independently from *Mike 11*.

All control features have to be programmed into the *UWC*. Water levels are read from the memory locations mapped by the *UWC* according to pointers provided by *Mike 11*. New gate positions are computed in the *UWC* according to the control calculations. The *UWC* must redefine the head–discharge relationship for gates as internal boundary conditions. These new gate positions are used to compute the solution coefficients which are inserted into the memory locations defined by the *UWC* mapping.

The *UWC* is set up as a dynamic link library (M11UWC.DLL). It is accessed by *Mike 11* from its computational engine. This file must be in the *Mike 11* root directory. Different versions of the *UWC* can be substituted into that directory without exiting the *Mike 11* program, since it is linked at simulation run time.

In order to control flow at check structures, the *UWC* must include *Mike 11* equations for gate hydraulics. Calculation of the desired gate opening to provide a desired flow requires either inversion of these equations, or a search procedure. The *UWC* can be made responsible for computing the turnout gate position changes needed to simulate changes in demand, otherwise the schedule has to be computed externally. Thus a routine for mimicking the gate settings of zanjeros has to be included in the *UWC*.

An important difference in the *Mike 11* interface is that it accesses the calculations at each iteration. This allows the *UWC* to overwrite the built-in head–discharge relationships for gates, and thus modify the system of finite difference equations.

Overall, development of the *UWC* was time consuming and difficult. As programmed, it can model a wide variety of canal topologies. It allows a great deal of flexibility in modeling control situations.

Sobek

Sobek has the ability to be linked to *MATLAB* (*Matlab users guide* 2000). The *Sobek* channel flow (CF) module is the unsteady open-channel flow simulation portion of *Sobek*. The *Sobek* RTC module allows the check gates in *Sobek* CF to be controlled externally by *MATLAB*. Automatic control algorithms can be written as *MATLAB* m-files that are then connected to the *Sobek* CF module through the *Sobek* RTC module. Within *Sobek*, the user determines that control is from an external source and selects the "m-file" that is to be used for control from a directory list.

The *Sobek* RTC passes the various hydraulic properties (i.e., water depths and flow rates) from *Sobek* CF to the controller code (i.e., m-file). Gate positions and water levels are available in *MATLAB* using ID names defined by *Sobek*. The controller code uses this information along with the information on the canal properties to calculate the appropriate adjustment to the individual check gate structures using *MATLAB*. Finally, this information is passed back to *Sobek* CF through *Sobek* RTC and the appropriate control actions are implemented.

Because *MATLAB* has a variety of built-in and share-ware control engineering methods, the development of control calculations is much simpler than the programming required for the "custom" routines in *CanalCAD* or the *UWC* for *Mike 11*. We commonly use *MATLAB* to design the control algorithms or to do the controller tuning. Thus it is a simple matter to implement these controllers in *MATLAB*. Yet, *MATLAB* is an interpreted language, and as such is much slower at making these calculations. For simple control routines, this time is insignificant compared to the simulation computation time. However, for on-line optimization, compiled languages have a decided advantage over *MATLAB*. *MATLAB* also has limited object-oriented features.

Information that can be passed from *Sobek* CF to *MATLAB* includes water level elevations at nodes, flow rate through reaches, gate openings of structures, simulation date, and simulation time. Other physical and hydraulic properties of the structures (e.g., gate width, invert elevation, contraction coefficient, etc.) cannot be passed from *Sobek* CF to *MATLAB*. Thus, this information must be included in the controller m-file as well as in *Sobek* CF. Care must be taken to assure that these parameters are the same in both locations. The user only has the ability to modify the gate positions of the structures; the user cannot overwrite the gate hydraulics of a structure. Like the other two programs, it is the user's responsibility to use the correct gate equation and flow regime so that the control from *MATLAB* produces the correct results during simulation.

The setup in *Sobek* CF for automatic control consists of creating a few text files that define which structures are to be controlled and which water level elevations and flow rates are to be available to *MATLAB*. No additional initialization is needed in *MATLAB*. The real difficulty of setting up a canal system for automatic control deals with bookkeeping. *Sobek* assigns an ID name to each node, reach, and structure in the model. These ID names are then used in the text file that defines the information that is to be passed to *MATLAB*. These ID names are assigned when the cross section, structure, etc. are first defined. The user can specify a name at that time, but once they are assigned, they cannot be changed. Thus, the user needs to develop a table to associate *Sobek*'s ID names with the appropriate gate positions, water level elevations, or flow rates. Without some care initially,

this can cause difficulty in viewing and saving results.

Overall, we found the application of canal automation in *Sobek* to be both flexible and straightforward. The ability to interface with *MATLAB* is a real step forward in the application of canal automation of a canal reach.

Further Needs for Canal Controller Simulation

Simple routines are needed for establishing initial steady-state conditions. This has been a problem for all three software packages. This problem is particularly difficult with zero flow at the downstream end.

Indexing and naming of structures and associated water levels cause confusion with all these software packages. User-defined naming that transitions from setup to simulation to control to output are needed to make development of control routines and interpretation of results more straightforward. The ability to define control segments and control different sections of the canal network with different logic would also be useful. We have done this to a certain degree with the *Mike 11 UWC* interface, but this has not yet been really well integrated.

The development and tuning of canal controllers generally requires hydraulic properties for each canal pool. Dynamic regulation (Deltour and Sanfilippo 1998) and the volume compensation method for routing known flow changes through a canal (Bautista and Clemmens 2005) require volume as a function of flow rate, downstream water level setpoint, and flow resistance (e.g., *Manning* n). To date, the steady-flow routines in HEC–RAS (HEC 1997) has been the most convenient tool for developing these relationships. It would be useful if these control-related software packages could make development of these relationships more straightforward.

For feedback control, other pool properties are needed that can typically best be determined through unsteady-flow simulation. We often use the integrator-delay model of Schuurmans et al. (1999). The properties of this model also vary with flow rate, downstream water level setpoint, and flow resistance (e.g., *Manning* n). These properties can be determined by providing a step increase in discharge at the upstream end and observing the water level response at the downstream end when holding the downstream outflow constant. This approach requires analysis of one pool at a time, typically requiring a new model for each pool with new boundary conditions. It would be convenient if this could be done somewhat automatically. System identification (Silvis et al. 1998) is another approach to determining canal pool properties. Difficulties with control of pool inflow and outflow in the middle of a canal make this more difficult and the resulting parameters less precise. Silvis et al. (1998) actually identified canal properties on a real canal, where separation of one pool from the canal is physically impossible. In simulation, we have the luxury of isolating individual pools for study.

Gate hydraulics has been a problem with all these models. It would be helpful if the user could specify head–discharge–gate–position relationships for a given gate. This would ensure that the same relationships were used in simulation and control. It might also avoid problems with the transitions between orifice and weir flow. We have had difficulties with these programs in being able to predict when the orifice is open far enough to change from orifice to weir flow. In theory, when the gate is more than $2/3$ of the upstream head the water can pass under the gate as orifice flow. In practice, the flow does not automatically switch flow regimes under these conditions. For control purposes, it is desirable not to pull the gate out of the water, because at that point the gate no longer controls the rate of flow. A user-defined relationship could easily avoid this complexity. The ability to access the gate hydraulic routines in the computational engine from the control routines would be an equally useful option. The *UWC* for *Mike 11* allows this, but this is still essentially independent from *Mike 11*.

Documentation of these programs is relatively good, although there are many hydraulic situations that are not well documented. Documentation of the controller interfaces is scanty at best. Documentation on how to set up these models to simulate canals under automatic controls is limited.

Support and Cost

The list price for *CanalCAD* was $3,000, with additional security keys available for $1,000 each. Support is somewhat limited due to the limited number of users. Most upgrades have been funded directly by the various users. The original programmer of the software, John Parrish, is the only one who can provide upgrades and fixes. No annual maintenance contracts are offered. It is unclear whether or not this program will be supported in the future.

The list price for *Mike 11* studio (*version 4*), which include river modeling with 250 nodes and structure operations is $1,995, with $1,500 annual maintenance. These costs are substantially less than a decade ago. Additional capabilities are available with companion software products. Training and support from DHI for routine modeling applications is good. This is a growing product line and future development and support is expected to be good. The *UWC* is fully supported in *version 4*.

Sobek is sold in the United States through XP-Software, with a current list price of $12,000 (United States only), with a 50% discount for educational versions. The annual maintenance contract with Delft Hydraulics is *$*4000/year. Support has been reasonably good. This is a relatively new product for Delft Hydraulics and augments a wider line of software products. Continued development and support are expected. (Delft Hydraulics 2000).

Example

These simulation models should provide the same results for a given situation. To verify this, we chose to run all three

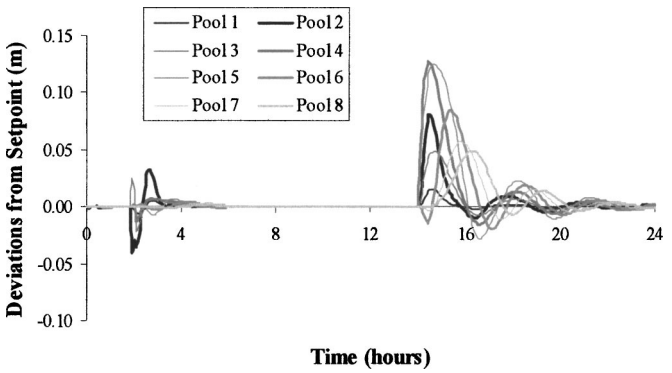

Fig. 1. Simulation results from CanalCAD for Test Case 1-1 (fully centralized proportional integral controller, untuned, no gate constraints, $\theta = 0.6$, $R_1 = 20$)

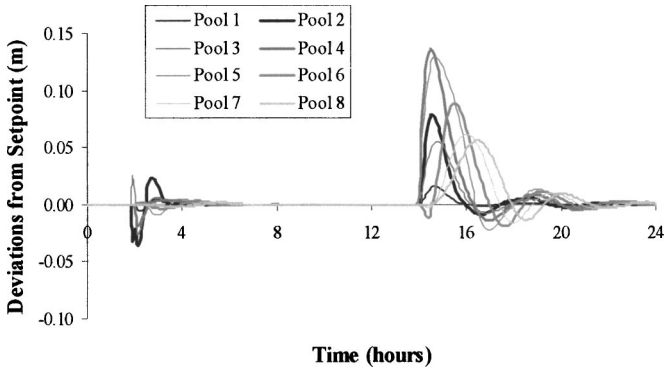

Fig. 2. Simulation results from *Mike 11* for Test Case 1-1 (fully centralized proportional integral controller, untuned, no gate constraints, $\delta = 0.6$, $R_1 = 20$)

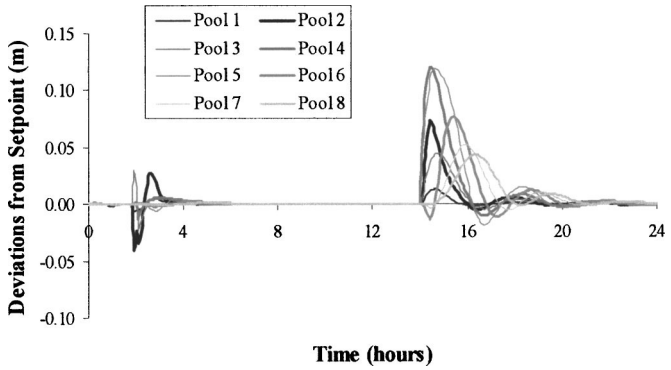

Fig. 3. Simulation results from Sobek for Test Case 1-1 (fully centralized proportional-integral controller, untuned, no gate constraints, $\theta = 0.6$, $R_1 = 20$)

simulation models with the same canal test and the same control algorithm. Test Case 1-1 from the ASCE canal control algorithms tests was chosen for comparison. Details for this test are given in Clemmens et al. (1998). Performance indicators include the maximum absolute error, integrated absolute error, steady state error, and the integrated absolute flow changes. These four performance indicators are to be provided as the maximum and average of all pools for two 12 h periods. For these tests we chose to use the feedback control algorithm described in Clemmens and Schuurmans (2004) and the feedforward algorithm described by Bautista and Clemmens (2005). These tests were run under untuned conditions, without constraints on gate movement, and with a fully centralized proportional integral controller designed under tuned conditions with an optimization penalty $R_1 = 20$. Details of the application of these controllers to the test cases and the resulting performance parameters are described in Clemmens and Wahlin (2004). All simulations were performed with the time-weighting parameter $\theta = 0.60$ ($\delta = 0.6$ in *Mike 11*). The check gate hydraulics are slightly different for *Mike 11*, as discussed earlier. However, this should have little impact on the simulations since these gates are free flowing and the control calculations adjust gate flows and not gate positions. A separate routine determines the gate position needed for the required flow.

The results from *CanalCAD*, *Mike 11*, and *Sobek* are shown in Figs. 1–3. Some differences exist, particularly about 14–16 h into the test. However, these differences are minor and may result from slight differences in how the controllers are implemented or in how damping is defined (i.e., θ versus δ). Since the interfaces, languages, etc. are totally different, there is no guarantee that the controller implementations are identical. The performance parameters are shown in Table 1. Again while some differences exist, these differences are minor and suggest that the simulation models predict similar hydraulic behavior.

Summary and Conclusions

In this paper, we examined three unsteady-flow simulation models that can be used to study canal operations and automation; *CanalCAD*, *Mike 11*, and *Sobek*. We chose these three programs since they were, at the time of our studies, the only three programs available to allow any user to write their own canal control routines. We found all three programs useful for studying various aspect of canal automation. *CanalCAD* has advantages in that it was developed for canal automation studies. For simple canals and theoretical studies it is the most convenient to use. However, it is not able to handle branching networks, supercritical flow, and other situations that occur in operating systems. *Mike 11* and *Sobek* are more comprehensive simulation packages that handle a wider range of conditions. However, this flexibility also causes greater complexity in setting up canals, running simulations, and interpreting output.

An example shows that all three models produce similar results, thus selection among them should be based on requirements for a particular study, as well as cost, convenience in data input, requirements for simulation testing, and data output. These are likely to change over time, as technology progresses. *Sobek* provides a real advantage over the other two programs in that it allows interface to *Matlab*, a technical computing environment.

All these programs had difficulties in establishing steady-state conditions. The particular hydraulic conditions represented by canal operations cause more difficulty than for a typical river

Table 1. Performance Parameters for ASCE Test Case 1-1, Untuned, No Gate Constraints, Fully Centralized Proportional Integrated Controller ($R_1 = 20$), Volume-Based Feedforward, $\theta = 0.6$

	Maximum absolute error (%)				Integrated absolute error (%)				Steady state error (%)				Integrated absolute flow changes (m^3/s)			
	0–12		12–24		0–12		12–24		0–12		12–24		0–12		12–24	
	Max.	Avg.	Max.	Avg.	Max.	Avg.	Max.	Avg.	Max.	Avg.	Max.	Avg.	Max.	Avg.	Max.	Avg.
CanalCAD	4.4%	1.4%	14.2%	8.6%	0.3%	0.1%	2.2%	1.2%	0.0%	0.0%	0.1%	0.1%	0.17	0.07	0.37	0.27
Sobek	4.5%	1.5%	13.4%	7.9%	0.3%	0.1%	1.9%	1.0%	0.0%	0.0%	0.1%	0.0%	0.18	0.07	0.40	0.20
Mike 11	3.9%	1.5%	15.2%	9.1%	0.3%	0.1%	2.3%	1.3%	0.0%	0.0%	0.1%	0.1%	0.12	0.05	0.30	0.21

Note: Max.=maximum and Avg.=average.

modeling application. Further work in providing more robust steady-state conditions would be useful for all three programs.

For future work in canal automation, these programs need to have a convenient method for determining canal properties from simple simulation tests. Currently, this requires developing a new canal model for each canal pool and running a series of tests on that new canal, developing new steady-state conditions for each test, etc. Once a canal model is developed, it would be convenient to be able to separate out one pool at a time for running tests on canal properties. This might require a considerable amount of development/programming work. Based on past needs, such an effort could probably not be justified. But as more automatic controls are implemented on canals, such an option will likely be useful in the future.

Overall, we found all three software packages useful for canal automation studies, with documentation and support consistent with cost. Hopefully the discussion provided here will help future software to more closely fit the needs for this application.

References

Abbott, M. B., and Ionescu, F. (1967). "On the numerical computation of nearly-horizontal flows." *J. Hydraul. Res.*, 5(2), 97–117.

American Society of Civil Engineers (ASCE). (1993). "Unsteady-flow modeling of irrigation canals." *J. Irrig. Drain. Eng.*, 119(4), 615–630.

Bautista, E., and Clemmens, A. J. (2005). "Volume compensation method for routing irrigation canal demand changes." *J. Irrig. Drain. Eng.*, in press.

Burt, C. M. (1983). "Regulation of sloping canals by automatic downstream control." PhD thesis, Utah State Univ., Logan, Utah.

Burt, C. M., and Gartrell, G. (1993). "Irrigation-canal—simulation model usage." *J. Irrig. Drain. Eng.*, 119(4), 613–742.

Burt, C. M., Mills, R. S., Khalsa, R. D., and Ruiz, C. V. (1998). "Improved proportional-integral (PI) logic for canal automation." *J. Irrig. Drain. Eng.*, 124(1), 53–57.

Clemmens, A. J. (1993). "Editorial on canal system hydraulic modeling." *J. Irrig. Drain. Eng.*, 119(4), 613–614.

Clemmens, A. J. (1998). "Editorial." *J. Irrig. Drain. Eng.*, 124(1), 1–2.

Clemmens, A. J., Bautista, E., and Strand, R. J. (1997). "Canal automation pilot project, Phase I Report." *WCL Rep. No. 22*, United States Water Conservation Laboratory, Phoenix, Ariz.

Clemmens, A. J., Holly, F. M., and Schuurmans, W. (1993). "Description and evaluation of program: Duflow." *J. Irrig. Drain. Eng.*, 119(4), 724–734.

Clemmens, A. J., Kacerek, T., Grawitz, B., and Schuurmans, W. (1998). "Test cases for canal control algorithms." *J. Irrig. Drain. Eng.*, 124(1), 23–30.

Clemmens, A. J., and Schuurmans, J. (2004). "Simple optimal feedback canal controllers: theory." *J. Irrig. Drain. Eng.*, 130(1), 26–34.

Clemmens, A. J., and Wahlin, B. T. (2004). "Simple optimal feedback canal controllers: ASCE test-case results." *J. Irrig. Drain. Eng.*, 130(1), 35–46.

Contractor, D. N., and Schuurmans, W. (1993). "Informed use and pitfalls of canal models." *J. Irrig. Drain. Eng.*, 119(4), 663–672.

Cunge, J. A., Holly, F. M., Jr., and Verwey, A. (1980). *Practical aspects of computational river hydraulics*, Pitman, Boston.

Danish Hydraulics Institute (DHI). (1995). *Mike 11 Version 3.11 user manual*, Horsholm, Denmark.

Delft Hydraulics (2000). *Sobek flow module technical reference guide, version 2.2*, Delft Hydraulics, Delft, The Netherlands.

Deltour, J.-L., and Sanfilippo, F. (1998). "Introduction of the Smith predictor into dynamic regulation." *J. Irrig. Drain. Eng.*, 124(1), 47–52.

Holly, F. M., Jr., and Merkley, G. P. (1993). "Unique problems in modeling irrigation canals." *J. Irrig. Drain. Eng.*, 119(4), 656–662.

Holly, F. M., Jr., and Parrish, J. B., III, (1992). "CanalCAD: dynamic flow simulation in irrigation canals with automatic control." *Limited Distribution Rep. No. 196*, Iowa Institute of Hydraulic Research, Univ. of Iowa, Iowa City, Iowa.

Holly, F. M., Jr., and Parrish, J. B., III (1993). "Description and evaluation of program: CARIMA." *J. Irrig. Drain. Eng.*, 119(4), 703–713.

Hydrologic Engineering Center (HEC). (1997). *HEC-RAS River Systems Analysis, Version 2, user's manual*, United States Army Corp of Engineers, Davis, Calif.

Lin, Z., and Manz, D. H. (1992). "Optimal operation of irrigation canal systems using nonlinear programming—dynamic simulation model." *Proc., International Workshop on The Application of Mathematical Modeling for the Improvement of Irrigation Canal Operations*, Cemagref, Montpellier, France.

Liu, F., Feyen, J., Malaterre, P.-O., Baume, J.-P., and Kosuth, P. (1998). "Development and evaluation of canal automation algorithm CLIS." *J. Irrig. Drain. Eng.*, 124(1), 40–46.

Malaterre, P.-O. (1998). "PILOTE: Linear quadratic optimal controller for irrigation canals." *J. Irrig. Drain. Eng.*, 124(4), 187–194.

Malaterre, P. O., Rogers, D. C., and Schuurmans, J. (1998). "Classification of canal control algorithms." *J. Irrig. Drain. Eng.*, 124(1), 3–10.

MATLAB user's guide, Version 6. (2000). The Math Works Inc., Natick, Mass.

Merkley, G. P., and Walker, W. W. (1991). "Centralized scheduling logic for canal operation." *J. Irrig. Drain. Eng.*, 117(3), 337–393.

Parrish, J. B., III, and Khalsa, R. D. (1997). "Calibration of open channel flow computer simulation." *Proc., 27th IAHR Congress, Theme A Managing Water: Coping with Scarcity and Abundance*, 338–342.

Rogers, D. C., Ehler, D. G., Falver, H. T., Serfozo, E. A., Voorheis, P., Johansen, R. P., Arrington, R. M., and Rossi, L. J. (1995). *Canal systems automation manual*, Vol. 2, United States Bureau of Reclamation, Denver.

Rogers, D. C., and Goussard, J. (1998). "Canal control algorithms currently in use." *J. Irrig. Drain. Eng.*, 124(1), 11–15.

Rogers, D. C., and Merkley, G. P. (1993). "Description and evaluation of the program USM." *J. Irrig. Drain. Eng.*, 119(4), 693–702.

Schuurmans, W. (1993). "Description and evaluation of program Modis." *J. Irrig. Drain. Eng.*, 119(4), 735–742.

Schuurmans, J., Clemmens, A. J., Dijkstra, S., Hof, A., and Brouwer, R. (1999). "Modeling of irrigation and drainage canals for controller design." *J. Irrig. Drain. Eng.*, 125(6), 338–344.

Silvis, L. G., Hof, A., van den Hof, P. M. J., and Clemmens, A. J. (1998). "System identification on open channels." *Proc., 3rd Int. Conf. on Hydroinformatics*, Copenhagen, Denmark, V. Babovic and L. C. Larsen, eds., Balkema, Rotterdam, The Netherlands, 21–224.

Stelleing, G. S., Kernkamp, H. W. J., and Laguzzi, M. M. (1998). "Delft flooding system: A powerful tool for inundation assessment based upon positive flow simulation." *Proc., 3rd Int. Conf. on Hydroinformatics*, Copenhagen, Denmark, V. Babovic and L. C. Larsen, eds., Balkema, Rotterdam, The Netherlands, 449–456.

Strelkoff, T. S., and Falvey, H. T. (1993). "Numerical methods used to model unsteady canal flow." *J. Irrig. Drain. Eng.*, 119(4), 637–7655.

van Overloop, P. J., Schuurmans, W., and Brouwer, R. (2003). "Model predictive control of canal systems in the Netherlands." *Proc., 2nd Int. Conf. on Irrigation and Drainage*, USCID, Denver.

Wahlin, B. T. (2002). "Optimal feedback controller design for branching canal networks." PhD thesis, Arizona State Univ., Tempe, Ariz.

Wahlin, B. T., and Clemmens, A. J. (2002). "Performance of several historic canal control algorithms on the ASCE test cases." *J. Irrig. Drain. Eng.*, 128(6), 365–375.

Zimbelman, D. D., and Bedworth, D. (1983). "Computer control for irrigation-canal system." *J. Irrig. Drain. Eng.*, 109(1), 43–59.